Eurographics/ IEEE-VGTC Symposium on Visualization (2006)
Thomas Ertl, Ken Joy, and Beatriz Santos (Editors)

Real-Time Super Resolution Contextual Close-up of Clinical Volumetric Data

T. Taerum[1] M. C. Sousa[1] F. Samavati[1] S. Chan[1,4] J. R. Mitchell[1,2,3,4] †

Departments of [1]Computer Science, [2]Radiology, [3]Clinical Neurosciences, University of Calgary, Canada
[4]Seaman Family MR Research Centre, Foothills Medical Centre, Calgary, Canada ‡

Abstract

We present an illustrative visualization system for real-time and high quality rendering of clinical volumetric medical data. Our technique is inspired by a medical illustration technique for depicting contextual close-up views of selected regions of interest where internal anatomical features are rendered in high detail. Our method integrates four important components: decimation of original volume for interactivity, B-spline subdivision for super-resolution rendering, fast gradient quantization technique for feature extraction and GPU fragment shaders for gradient dependent rendering and transfer functions. Examples with clinical CT and MRI data demonstrate the capabilities of our system.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: I.3.3 Picture/Image Generation J.3 [Computer Applications]: Life and Medical Sciences

1. Introduction

Clinical applications often require effective visualization of internal features of 3D medical images. Rapid determination of the size, shape, and spatial location or proximity of a lesion, for instance, could be critical in some situations. To properly estimate these elements, clinicians may need to be able to visualize the data from multiple viewing directions and have a high-quality, clear image of a specific internal region of interest, while preserving overall contextual and spatial information. Existing techniques for direct volume rendering often suffer from the problem of occlusion by exterior features. Transfer functions, clipping or segmentation can alleviate this problem; however these techniques typically obscure details in the final image with overlapping structures and also remove important contextual information [BGKG05]. Ideally a user should be able to see an internal *Region of Interest* (hereafter, ROI) while still receiving contextual feedback. In this paper, we present an exploration tool that provides real-time high quality direct volume rendering while maintaining context, on commercial off-the-shelf graphics processing units (GPU) commonly available in inexpensive personal computers. We were inspired by a particular technique used by medical illustrators to depict *contextual close-up* views as shown in Figure 1. In this technique, the enlarged and separated image (i.e. a circular or rectangular ROI) provides a clear close-up view and focus of attention on the internal features while still maintaining a very strong indication of context with respect to the subject in question.

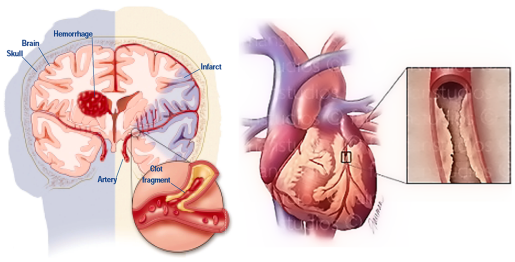

Figure 1: *Traditional medical illustrations using contextual close-up view technique depicting internal features with full context still visible. Illustrations printed with permission: Copyright Fairman Studios, LLC 2005.* http://www.fairmanstudios.com

† http://www.ImagingInformatics.ca
‡ http://www.mrcentre.ca

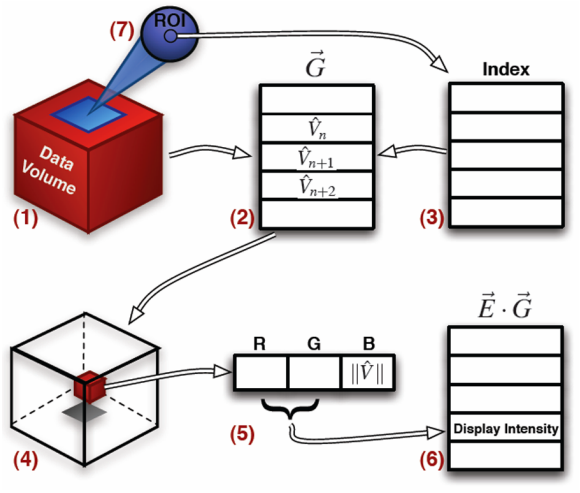

Figure 2: *Seven stages of our volume rendering pipeline.*

2. System overview

Our method integrates four key components to provide real-time feedback and high quality rendering approximating the traditional illustration technique of contextual close-up: **(a)** a decimated lower resolution of the volume for interactivity; **(b)** B-spline subdivision to generate a higher resolution sub-image for the ROI; **(c)** a fast dual-access gradient quantization technique for silhouette enhancement and Phong shading; **(d)** GPU shaders for gradient dependent rendering and transfer functions. These four components are processed in seven key steps (Figure 2). **Step (1)** The data volume is loaded and pre-processed with a fast B-spline multi-resolution reverse subdivision technique, resulting in a decimated copy of the volume for fast interaction with it. Gradients $\vec{V}$are then calculated and quantized for each voxel. A highly optimized quantization algorithm ensures rapid quantization, this requires only a few seconds to complete. **Step (2)** Quantized gradients from the main data volume are stored in a short "static" primary lookup table. Indices into this table are used to construct the quantized gradient volume (steps 4 and 5). **Step (3)** A secondary gradient table is constructed with indices pointing to entries in the primary gradient table (step 2). This secondary table is used to access an existing quantized gradient in the primary table, given any arbitrary new gradient vector coming from the super-resolution ROI (step 7). **Step (4)** The index into the quantized gradient table for each data voxel is stored in a 3D array. This array has the same dimensions as the data volume. The array is uploaded to the GPU and stored as a 3D texture. This allows rapid access to gradient information on a per-voxel basis within a GPU fragment program. **Step (5)** We store gradient information in each 3D texture element as an *RGB* value. The *R* and *G* components hold the index into the quantized gradient table. This index is used to access the "display intensity table" (step 6). The *B* component contains the magnitude of the quantized gradient for the corresponding data voxel. The gradient magnitude is used for other transfer functions. **Step (6)** The display intensity table is a list of greyscale values. This table is the same length as the quantized gradient table in step 2 (thus, the same set of indices can be used to index into either table). Table elements are determined by calculating the inner product between the current view (or lighting) vector, and the quantized gradient values from step 2. This table must be updated each time the view is changed. However, this operation is very fast (typically less than 1 ms) since the quantized gradient table is short. **Step (7)** During run-time, contextual close-up views are created. A 64^3 sub-volume ROI is defined inside the main data volume. B-spline subdivision of the ROI yields a 128^3 super-resolution sub-volume that is rendered enlarged and separated from the sub-volume. The secondary gradient indexing method (step 3) allows for real-time gradient dependent rendering of the ROI.

Contributions The main contribution of this paper is a direct volume *illustrative visualization* system that provides interactive high-quality visualization on inexpensive GPUs, along with interactive visualization of contextual close-ups (Figure 1), a technique traditionally used by medical illustrators. Innovations with our technique include: (1) Using a fast B-spline multi-resolution subdivision technique to generate both a low-resolution representation of the volume for fast interactive frame rates (Section 4), and a smooth super-resolution representation of the volume for high quality visualization (Section 5). We also use the B-spline subdivision to generate a smooth super-resolution representation of the gradients (Section 5). This requires a gradient data structure that has both efficient time and memory complexity. We introduce (2) a new gradient quantization method and data structure (Section 6) that has very fast vector quantization and the ability to quickly access a quantized vector with either an index or an arbitrary vector, and provides approximately a 6:1 memory savings for gradients. This data structure also allows for very fast evaluation of inner products required for shading and silhouette extraction. Additionally, (3) this data structure is easily integrated into a GPU fragment shader to alleviate some of the computational load from the CPU.

The rest of this paper is organized as follows: Related research is reviewed in Section 3. Details of our approach are provided in Sections 4 - 7. Results are discussed in Section 8, and conclusions and future work presented in Section 9.

3. Related work

We review existing research works within four categories.

(1) Multi-resolution for interactive volume manipulation: Weiler et al. [WWH*00] present a multi-resolution technique for volume rendering that includes rendering using multiple levels of detail at the same time. The original volume is subdivided into small bricks and during rendering

each brick is assigned a level of detail based on its distance from a focus point oracle. Our technique differs in that we reduce the resolution of the entire dataset by one level for use during interactivity rather than maintaining a brick data structure. LaMar et al. [LHJ99] deal with exploration of very large volume datasets. The level of detail rendered is determined interactively by the user. If a user wishes more detail of a specified block a higher resolution representation of the data is presented for that block. Their technique also informs the user of the error incurred at a specific level of detail, to help guide the user in level of detail selection. In our technique we deal with smaller datasets and provide an intuitive tool for visualizing internal features at a super-resolution. Their technique uses large datasets and allows the user to view selected sub-volumes at various levels of detail. In [WS05] Wang and Shen present a technique in which the 3D texture is divided into bricks with hierarchical levels of detail. During the rendering phase the level of detail used for a brick is determined by distance to the viewer and field of view. They also make use of spherical shells for their proxy geometry. Again, this technique uses the brick data structure to allow various levels of detail rather than as in our approach which uses decimation of the entire volume for interactivity.

(2) Volumetric gradient dependent rendering: Many techniques have been proposed for silhouette extraction and rendering using: trivariate B-spline tensor [SE04], hardware based method for continuous one-pixel width silhouette [NK04], and GPU fragment shaders for feature halos near to silhouette enhancements [SE03]. Csebfalvi et al. [CLMAK01] use a lookup table of quantized gradient direction for fast evaluation of inner product calculations required for gradient dependent rendering. In our approach we have a dual access lookup table for finer gradients computed at the ROI. More recently, Burns et al. [BKR*05] presented a technique to quickly extract line contours using seed point searching and exploiting spatio-temporal coherency. The primary issue with volumetric gradient dependent rendering is the evaluation of the inner products between the gradient and other directions (i.e. viewing, lighting), which can be minimized by using gradient quantization techniques. Gallinger [Gal02] analyzes several gradient quantization techniques for their effect on image quality for volume rendering. His results clearly indicate that it is rarely necessary to represent a 3D gradient using the full precision available with floating point numbers. Most gradient quantization methods involve generating a codebook of gradients and storing indices into the quantized gradient codebook. Issues with quantization involve generation time, storage benefits, gradient error, and retrieval time.

(3) Lenses for data exploration: Our ROI relates to previous methods for exploring and visualizing data using the metaphor of lenses. Bier et al. [BSP*99] introduce the 'magic' lenses approach, later extended by Viega et al. [VCWP96] to 3D volumetric lenses. Shaw et al. [SHER99] use x-ray lenses to visualize important features of a volume dataset by culling user specified intensity values from rendering inside the lens. LaMar et al. [LHJ01] show how to ensure smooth blending for display between areas inside their 3D lens and areas outside the lens, thus improving the visualization by avoiding discontinuity artifacts. Zhou et al. [ZHT02] combine non-photorealistic rendering (NPR) with focal region based rendering to draw attention to specific areas of interest. Most recently, Wang et al. [WZMK05] presented a volume lens technique using ray casting. The rays are refracted at the image plane based on the lens chosen. The GPU fragment program then steps through the 3D texture compositing a fragment color based on texels encountered along the ray.

(4) Illustration-based focus of attention: Recently, researchers have proposed techniques inspired by traditional technical, medical and scientific illustration to create contextual focus of attention. Viola et al. [VKG04] present a technique called *maximum importance projection*, applied to pre-segmented volume data, where a user selects an important range of intensities, and the system renders materials deemed important with full opacity and less important materials transparent. Svakhine and Ebert [SES05] present a system in that GPU fragment programs are used to apply transfer functions to achieve illustrative results with direct volume rendering. Bruckner et al. [BGKG05] introduce a ray casting-based technique with an automated opacity function. This allows a user to simply choose a color for an intensity range rather than having to struggle with determining a suitable opacity to assign for the transfer function. The opacity function is based mostly on intensity, gradient magnitude, depth, and a Phong-Blinn lighting model. Additionally they provide clipping planes to view internal features. Bruckner and Gröller [BG05] describe a system where sub-volumes are interactively selected, extracted and rendered to a separate area of the screen, allowing visualization of internal features. The primary difference between our technique and this is our use of B-splines to perform super-sampling to the sub-volume prior to rendering; additionally, B-spline super-sampling is applied to the sub-volume's gradient vector field and our new data structure for gradient quantization allows us to make use of quantized gradients approximated from the super-resolution gradient vector field.

4. Volume representation

Medical image datasets can be very large. For instance, a CT scan of the head can have spatial dimensions up to $512x512x256$ voxels, while an MRI dataset can be as large as 256^3 voxels. 16-bit samples are commonly produced in both modalities. Although many graphics display devices are now equipped with enough memory to store the entire volume, few commodity devices are actually capable of rendering the full resolution volume with the desired quality and enhancements of transfer functions at interactive rates. In order to ensure that the data can be viewed and manipulated at

interactive rates, while still allowing for full resolution viewing and detailed high resolution exploration of the data, we use a B-spline tensor at multiple resolutions to represent the data. Unser [Uns97,Uns02] describes several cost benefit advantages for using splines to model images. For our system, since the ROI is rendered enlarged (i.e. close-up view area as in Figure 1), a higher resolution image is required using some form of interpolation. Tri-linear interpolation is performed automatically by the graphics hardware, however we desired better quality. Unser [Uns97] describes that B-spline smoothing functions can provide quality superior to linear interpolation. However, rather than evaluating the B-spline basis functions, we make use of B-spline subdivision since it is very fast and also makes it easy for us to maintain an efficient hierarchical multi-resolution structure. The full B-spline representation of a trivariate dataset is as follows:

$$L(u,v,w) = \sum_{i=0}^{p-1}\sum_{j=0}^{q-1}\sum_{k=0}^{r-1} V_{ijk}B \qquad (1)$$

where V_{ijk} are the intensities from the image with dimensions $p \times q \times r$, and $B = B_i^m(u)B_j^m(v)B_k^m(w)$ are the basis functions for B-spline. In this setting, intensity values are smoothly blended using B-spline basis functions. u, v, and w show the parameter for the x, y, and z axis, respectively. Consequently, $L(u,v,w)$ is a continuous representation for the discrete dataset V_{ijk}. Functional evaluation of equation 1 is not fast enough for our purpose. Instead, we use subdivision algorithms for fast evaluation of equation 1, in particular the Chaikin subdivision algorithm, which can rapidly generate a quadratic B-spline:

$$c_{2i}^{k+1} = \tfrac{1}{4}c_{i-1}^k + \tfrac{3}{4}c_i^k, \quad c_{2i+1}^{k+1} = \tfrac{3}{4}c_i^k + \tfrac{1}{4}c_{i+1}^k,$$

where k and $k+1$ refer to the coarse and fine datasets and $c_{i-1}c_i$, c_{i+1} denote three consecutive voxel values along one of the main directions. Applying this scheme independently over all main directions x, y and z results in a higher resolution image that we use as a super-resolution representation (Figure 3). To obtain real-time interactivity, we use a low resolution approximation of the image. In order to have a consistent framework with our B-spline representation, we employ reverse subdivision rules [BS00]. We use the reverse Chaikin in our system:

$$c_i^k = -\frac{1}{4}c_{2i-1}^{k+1} + \frac{3}{4}c_{2i}^{k+1} + \frac{3}{4}c_{2i+1}^{k+1} - \frac{1}{4}c_{2i+2}^{k+1}, \qquad (2)$$

where $k+1$ refers to the image and k to the low resolution approximation. For an image N^3, equation 2 is applied in all three main directions to obtain an image $\left(\frac{N}{2}\right)^3$. Clearly the graphics hardware can render an image of this size much faster than the original image. Thus, the low resolution image is rendered any time the user is interacting with the system and upon cessation of interaction the system returns to the original resolution (Figure 3). This low resolution image is explicitly stored instead of evaluated on the fly to ensure the transition is very fast since it only requires a switch to the low resolution texture.

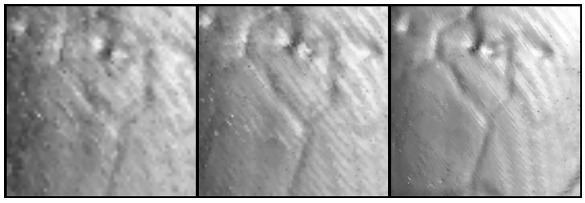

Figure 3: *A close-up view of a baby skull. (Left) low resolution volume used during user interaction. (Middle) the original resolution volume used upon cessation of user interaction. (Right) the super-resolution volume used in the ROI.*

5. Contextual close-up

With our technique, in order to visualize and explore the internal features of the dataset, a 3D position inside the volume is stored as the center of the ROI. A size for the ROI is also stored to define a box shaped sub-volume. The user is free to move this ROI to anywhere within the volume, simply with overloaded mouse controls. As the ROI is moved the sub-volume defined by the position of the ROI and the size of the box is updated.

To approximate the technique of medical illustrators to achieve both focus of attention and global context preservation for that focus (Figure 1), both the original image and the ROI need to be rendered simultaneously with some form of connection. The original image is rendered with the ROI fully exposed. This is achieved with a simple shape in the stencil buffer. The texture containing the super-resolution ROI is rendered enlarged in a circular or rectangular window shape, and in a different area of the screen. Then a translucent cone connecting the internal ROI and the up-sampled ROI is rendered. This results in the ROI being visible both at its physical location in the original volume image and in a separated screen area that draws attention to it (see Figures 7 and 8). The separated ROI is typically not occluded by anything since the texture consists only of the sub-volume defined by its position and size. Context is still available since the original image is rendered in full. The translucent cone provides a visual cue that connects the enlarged ROI to where it exists spatially inside the full 3D image. Since the user is able to move the ROI to anywhere inside the volume we use the luminance of the translucent cone for depth cueing.

Super Resolution ROI As mentioned before, since the ROI sub-volume is rendered enlarged, a form of up sampling is required. B-spline subdivision, as described in the previous section, is applied to the data over the ROI sub-volume to create a smooth high resolution representation of the ROI. The option to choose the order of the B-spline is provided to allow for different levels of smoothness (Figure 4).

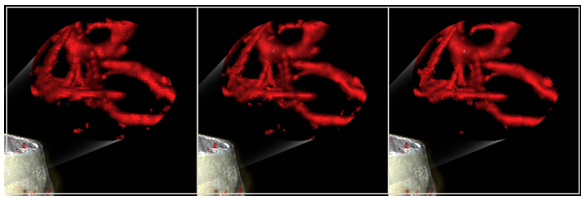

Figure 4: *Different levels of smoothness at the super-resolution ROI. (Left) no subdivision, (middle) order 3 and (right) order 4 subdivisions.*

When the ROI has the B-spline subdivision applied, in order to still have gradient dependent rendering techniques available, we need a way of quickly determining finer gradients in the ROI. One option would be to simply evaluate the gradients via central differences. However, this requires a square root operation to normalize the vector. A faster technique is possible. We apply the same B-spline subdivision algorithm used for the scalar image values, to the existing, already normalized gradients. Because B-spline subdivision has the property of unit summation, the vectors resulting from the subdivision will already be normalized. The next section describes how these new gradients from the ROI are processed.

6. Gradient quantization

Gradient dependent rendering techniques are important for direct illustrative volume rendering. The two techniques that are of primary concern for our system include silhouette enhancement and shading.

Extraction and rendering a **silhouette** is a valuable feature for a volumetric image, especially when dealing with internal features. Since volume data is very large, attempting to visualize the entire volume at once can be visually overwhelming. Silhouettes provide a trade-off that is ideal for volume rendering since it depicts the most information (i.e. important shape features) while rendering the least. In a volume, a voxel *v* is labelled a *silhouette* voxel if $(\vec{E} \cdot \vec{G}) = 0$, where $\vec{E}$ is the view vector, and $\vec{G}$ is the voxel intensity gradient. **Shading** with direct volume rendering implies having the luminance of a voxel determined via a shading model calculation. It is important in that it can provide the user with a basic indication of shape. The shading requires the calculation of $(\vec{L} \cdot \vec{G})$ where $\vec{L}$ is light vector for a directional light source.

As described previously in Section 3, the primary issue with volumetric gradient dependent rendering is the evaluation of the inner products between the gradient and other directions (i.e. viewing, lighting), which can be minimized by using gradient quantization techniques. Our goal is to design a gradient quantization technique that requires minimal generation time, has excellent storage benefits, minimal error, and fast retrieval time. In addition, in order to take full

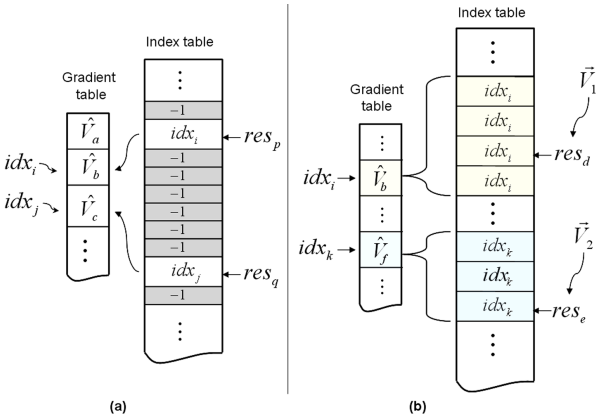

Figure 5: *(a) The gradient and index lookup tables constructed after algorithm QuantizeGradient(). (b) The gradient lookup table format, where a quantized gradient can be accessed via an index or an arbitrary vector ($\vec{V}_1$ and $\vec{V}_2$ in the figure).*

advantage of the gradient quantization while rendering the ROI, we require a special feature, the ability to access a quantized gradient with an arbitrary normalized vector, very quickly. We propose a dual access gradient quantization data structure as a solution to this. Our quantization technique has a very fast pre-processing stage, and provides immediate access to a quantized gradient using either an index or an arbitrary vector.

For this method we build two lookup tables (Figure 5): the *gradient* table, an array of quantized gradient vectors $\hat{V}$, and the *index* table, an array of integers, whose elements are an entry to the gradient table. The *index* table is used to access a quantized gradient with an arbitrary vector. To build these two lookup tables, as the gradient of each voxel is calculated and normalized, we apply a quantization function to the gradient $\hat{G}$, as described by the following algorithm:

```
QUANTIZE-GRADIENT(Ĝ)
1  V⃗.(x,y,z) ← round(Ĝ.(x,y,z) × γ)
2  res ← (V⃗.x + γ) << (b × 2) + (V⃗.y + γ) << (b) + (V⃗.z + γ)
3  if index[res] = −1
4    then idx ←next entry in gradient table
5         gradient[idx] ← V̂
6         index[res] ← idx
7         return idx
8  return index[res]
```

In *QuantizeGradient()*, $\hat{G}$ is the incoming gradient vector, γ is a small integer scale factor chosen at compile time and b is the minimum bits required to represent $(\gamma \times 2) + 1$. The incoming gradient $\hat{G}$ has components x, y and z in the range $[-1.0, 1.0]$. After Line 1, the vector $\vec{V}$ has components x, y and z in the range $[-\gamma, \gamma]$. Line 2 results in $res \in [0, 2^{3b}]$. The new vector $\vec{V}$ is only normalized and added to the gradient table (Line 5) if the value *res* has not been encountered yet

(Line 3). The position at which $\hat{V}$ is added is stored in the index table at *res* (Line 6 and Figure 5(a)). The index for the quantized gradient, whether a newly inserted one or not, is then returned (Lines 7 and 8). The value for γ determines the trade-off between quality and speed. Table 1 illustrates the results for various values of γ. In our experiments, we observed that a value of 31 for γ results in very fast access time, and a reasonably small size for the gradient table. This ensures that very few inner products for gradient dependent rendering are required. Additionally, the indices only require 2 bytes.

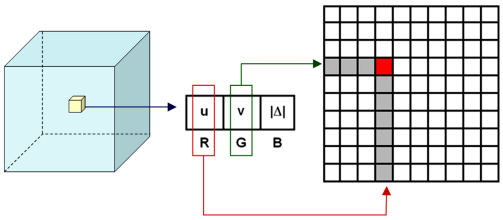

Figure 6: *For each voxel in the 3D texture, the silhouette or shading intensity is indexed via a texture indirection. Only the small 2D intensity texture is updated with each view change. The gradient magnitude $|\Delta|$ is used for other transfer functions.*

For each voxel, the index returned by *QuantizeGradient()* is stored in a 3D RGB texture. Since the index requires 2 bytes with $\gamma = 31$, it is stored such that the red component contains the high order byte and the green component contains the low order byte (Figure 6). Once every voxel gradient has been quantized, the result is a relatively small table of unique quantized gradients (Figure 5 (a)) and a slightly larger, however not full, table of indices pointing to entries in the gradient table (Figure 5 (b)). Additionally, each voxel contains a 2 byte index into the gradient table (Figure 6).

6.1. Primary indexing

To make use of the quantized gradient table for fast gradient dependent rendering, each time the view changes we iterate over the small gradients table and calculate a scalar intensity. For the silhouette extraction and enhancement, we use the following equation:

$$I_{silhouette} = 1 - abs(\vec{E} \cdot \vec{G})^{\alpha} \tag{3}$$

where $\vec{E}$ and $\vec{G}$ denote the eye and the gradient vectors, respectively, and α is the sharpness of the silhouette. For Phong shading, we place a directional light source at the eye (i.e. $\vec{L} = \vec{E}$) and compute the following equation:

$$I_{Phong} = abs(\vec{E} \cdot \vec{G}) + abs(\vec{R} \cdot \vec{G})^{n} \tag{4}$$

where n denotes the Phong shininess factor and the reflect vector $\vec{R} = 2\vec{G}(\vec{G} \cdot \vec{E}) - \vec{E}$. Using the view vector $\vec{E}$ for the light direction ensures the brightest areas are directly in front of the viewer. We found this approach provides the best results. Also notice that we use a directional light source instead of a point light source. A GPU shader program is used to perform the gradient dependent rendering. The results from evaluating both equations 3 and 4 are stored in a 256×256 texture. The results of silhouette evaluation are stored at an offset. As discussed in the previous section, each voxel in the 3D RGB texture has an index into the quantized gradient table stored with red being the high order byte and green being the low order byte. This translates perfectly into (u,v) texture coordinates for the gradient dependent rendering intensity texture. We also store the gradient magnitude in the same 3D texture for application of other transfer functions. The GPU fragment program then simply performs a texture indirection for each voxel to retrieve the silhouette or shading intensity. The results from the silhouette calculation are stored at an offset so that if the user selects the silhouette function the GPU shader program simply adds this offset to the u coordinate. This allows a user to select both silhouette enhancements of one material and shading for another.

6.2. Secondary indexing

The ROI is where our requirement for a secondary gradient indexing method comes in. Since we make use of the fast silhouette and shading technique described in the previous aubsection, gradients for each voxel are stored as an index rather than a vector. As previously discussed (Section 5), when the ROI has the B-spline subdivision applied, we also subdivide the gradients. In order to still use the fast silhouette and shading technique, we need a way of quickly determining an index for each of the new finer gradients into the quantized gradient table (Figure 5 (b)). Once the new finer gradients have been determined, the same quantization function from Lines 1 to 2 of *QuantizeGradient()* are applied to each new finer gradient to determine a value *res*. Recall however that (1) the intensity texture actually requires the index at which *res* is stored and not the actual value *res*; (2) we also inserted the index at which *res* resides in the index table at *res* (Figure 5 (a)).

An additional step is actually taken in the pre-process stage after the quantized gradient table is built: the index table is completely filled by inserting the closest existing index into each slot (Figure 5, (b)). Since the value of *res* from the gradient quantization function actually represents the full quantized gradient in a packed integer format, the resulting index table is, in a sense, sorted. By filling the table with the closest existing indices, that allows us to simply apply the quantization function to any arbitrary 3D vector and find the closest (most similar) existing vector in the quantized table and retrieve the quantized vector's index very quickly. Figure 5 (b) illustrates accessing a quantized gradient using indices or arbitrary vectors.

Scale (γ)	gradient table size	index table size	mean error	maximum error	access time
4 - 7	314 - 938	4096	8.95e-3 - 3.14e-3	6.15e-2 - 2.38e-2	125ms
8 - 15	1250 - 4249	32768	2.54e-3 - 7.33e-4	1.86e-2 - 5.48e-3	125ms
16 - 31	4874 - 17974	262144	6.29e-4 - 1.63e-4	5.09e-3 - 1.39e-3	125 - 172ms
32 - 63	19128 - 72381	2097152	1.55e-4 - 3.89e-5	1.28e-3 - 3.38e-4	281 - 407ms
64 - 127	75060 - 284599	16777216	3.73e-5 - 9.57e-6	3.37e-4 - 8.18e-5	468 - 562ms
128 - 255	289284 - 1072340	134217728	9.45e-6 - 2.34e-6	8.39e-5 - 2.05e-5	640 - 797ms

Table 1: *Gradient quantization results. Error values are $cos(\theta)$ between the incoming vector and the returned vector from the quantized gradient table (Figure 5). Access times are for performing 2 million lookups using a vector (the typical size of our ROI).*

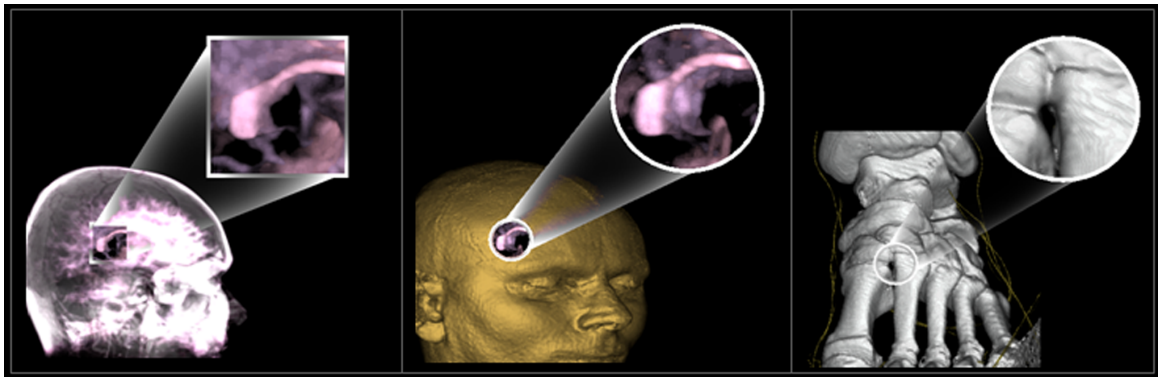

Figure 7: *Contextual close-up with circular and rectangular ROIs: MRI of brain with maximum intensity projection; CT of foot joints.*

7. Rendering pipeline

The rendering pipeline of our system has five main steps: **(1)** updating the ROI 3D texture using B-spline subdivision any time the ROI is moved; **(2)** updating the texture that stores the results of both equations 3 and 4 any time the viewing direction changes; **(3)** rendering the original 3D image as a 3D texture; **(4)** rendering the ROI as a 3D texture; **(5)** applying the desired transfer function to the two 3D textures.

The transfer function used to apply color is a 2D transfer function that employs the 2D histogram of intensities versus gradient magnitudes. Users are provided with the ability to select an area on the 2D histogram and designate a color. When this is performed, a 2D RGBA texture representing the entire area of the 2D histogram is updated by setting the color for the texels in the user selected area to the designated color. During rendering, the GPU fragment shader program performs a texture indirection using the intensity and gradient magnitude of the current voxel as texture coordinates to retrieve a color from the 2D RGBA texture.

8. Results and discussion

Our system achieved fast computations and interactive rates in the processes aiming at: (1) providing clear, enlarged, super-sampled images of internal volumetric regions of interest; (2) allowing the user to freely move and visualize this region to anywhere in the volume and with any orientation. Figures 7 and 8 (color plate) show results using our system in six volumetric datasets. All results were generated on an AMD Athlon 2500 with 1.25 GB of RAM and using OpenGL/ATI Radeon 9550 graphics card. Results for performing B-spline subdivision of the ROI requires $< 100ms$ for a region 64^3 subdivided to 128^3, thus allowing interactive frame rates to be easily achievable. Table 1 shows the results for using our dual access gradient quantization data structure. With a selection of $\gamma = 31$ access times are very fast, and the error incurred from using a quantized gradient as opposed to the original gradient is very small. When comparing our technique to performing inner products of a similar dataset for gradient dependent rendering directly in the GPU, our results indicate that our technique ranges from $5-10$ times faster. Also, we get a $6:1$ compression rate (2 bytes to encode the gradient instead of 12 bytes for 3 floats) with a small memory overhead of $< 700K$. In addition, the quantized gradients allow very fast mathematical evaluation required for gradient dependent rendering. Finally, frame rates for a dataset 256^3 range from 3-15 fps, depending on the transfer function chosen, which determines the number of texture indirections required.

9. Conclusions and future work

We presented an illustrative volume visualization system that approximates a medical illustration technique for depicting contextual close-up views of anatomical parts. Our system allows the user to interactively select, move and visualize regions of interest (ROI) anywhere within volumetric datasets. The system renders the contents of the ROI as super-resolution close-up views, while ignoring occluding features and preserving the global context of the visualization. Our method integrates the following important components: B-spline multi-resolution for low-resolution fast interaction, and super-resolution viewing of ROI, a fast gradient quantization technique for gradient dependant rendering and GPU fragment shaders for transfer functions. We plan to perform formal evaluation studies with clinicians and medical illustrators to determine the full effectiveness and illustrative capabilities of our system, and continue to investigate methods to approximate other traditional medical illustration techniques and styles.

Acknowledgements

We would like to thank the anonymous reviewers for the important and constructive comments and suggestions received. This work is being supported in part by Discovery Grants from the Natural Sciences and Engineering Research Council of Canada, iCORE and by awards from the Multiple Sclerosis Society of Canada, and the Alberta Heritage Foundation for Medical Research. We also thank Calgary Scientific Inc. (http://www.calgaryscientific.com/) for the partnership on integrating and extending our system.

References

[BG05] BRUCKNER S., GRÖLLER M. E.: Volumeshop: An interactive system for direct volume illustration. In *Proc. of IEEE Visualization '05* (2005), pp. 671–678.

[BGKG05] BRUCKNER S., GRIMM S., KANITSAR A., GRÖLLER M. E.: Illustrative context-preserving volume rendering. In *Proc. of EuroVis '05* (2005), pp. 69–76.

[BKR*05] BURNS M., KLAWE J., RUSINKIEWICZ S., FINKELSTEIN A., DECARLO D.: Line drawings from volume data. *ACM Transactions on Graphics (Proc. of SIGGRAPH '05) 24*, 3 (2005), 512–518.

[BS00] BARTELS R. H., SAMAVATI F. F.: Reversing subdivision rules: Local linear conditions and observations on inner products. *Journal of Computational and Applied Mathematics 119*, 1-2 (2000), 29–67. http://dx.doi.org/10.1016/S0377-0427(00)00370-8.

[BSP*99] BIER E., STONE M., PIER K., BUXTON W., DEROSE T.: Toolglass and magic lenses: The see-through interface. *Proc. of SIGGRAPH '93* (1999), 73–80.

[CLMAK01] CSEBFALVI B., LUKAS MROZ H. H., ANDREAS KONIG E. G.: Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum (Proc. of Eurographics '01) 20*, 3 (2001), 452–460.

[Gal02] GALLINGER I. J.: *Effects of Gradient Quantization on Quality of Volume-Rendered Images*. Master's thesis, School of Computing Science, Simon Fraser University, 2002.

[LHJ99] LAMAR E., HAMANN B., JOY K.: Multiresolution techniques for interactive texture-based volume visualization. In *Proc. IEEE Visualization '99* (1999), pp. 355–543.

[LHJ01] LAMAR E., HAMANN B., JOY K.: A magnification lens for interactive volume visualization. *Proc. of Pacific Graphics '01* (2001), 223–232.

[NK04] NAGY Z., KLEIN R.: High-quality silhouette illustration for texture based volume rendering. *Proc. of WSCG '04 12*, 1-3 (2004), 301–308.

[SE03] SVAKHINE N., EBERT D.: Interactive volume illustration and feature halos. In *Proc. of Pacific Graphics '03* (2003), pp. 347–354.

[SE04] SCHEIN S., ELBER G.: Adaptive extraction and visualization of silhouette curves from volumetric datasets. *The Visual Computer 20*, 4 (2004), 243–252.

[SES05] SVAKHINE N., EBERT D. S., STREDNEY D.: Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications 25*, 3 (2005), 31–39.

[SHER99] SHAW C., HALL J., EBERT D., ROBERTS A.: Interactive lens visualization techniques. *Proc. of Visualization '99* (1999), 155–160.

[Uns97] UNSER M.: Ten good reasons for using spline wavelets. In *Proceedings of the SPIE Conference on Mathematical Imaging: Wavelet Applications in Signal and Image Processing V* (1997), vol. 3169, pp. 422–431.

[Uns02] UNSER M.: Splines: A perfect fit for medical imaging. In *Progress in Biomedical Optics and Imaging, vol. 3, no. 22* (2002), vol. 4684, Part I of *Proceedings of the SPIE International Symposium on Medical Imaging: Image Processing (MI'02)*, pp. 225–236.

[VCWP96] VIEGA J., CONWAY M., WILLIAMS G., PAUSCH R.: 3d magic lenses. *Proc. of UIST '96* (1996), 51–58.

[VKG04] VIOLA I., KANITSAR A., GROLLER M.: Importance-driven volume rendering. *Proc. of Visualization '04* (2004), 139–145.

[WS05] WANG C., SHEN H.-W.: Hierarchical navigation interface: Leveraging multiple coordinated views for level-of-detail multiresolution volume rendering of large scientific data sets. In *Proc. of International Conference on Information Visualisation '05* (2005), pp. 259– 267.

[WWH*00] WEILER M., WESTERMANN R., HANSEN C., ZIMMERMAN K., ERTL T.: Level-of-detail volume rendering via 3d textures. In *Proc. IEEE Volume Visualization and Graphics Symposium '00* (2000), pp. 7–13.

[WZMK05] WANG L., ZHAO Y., MUELLER K., KAUFMAN A.: The magic volume lens: An interactive focus+context technique for volume rendering. In *Proc. of IEEE Visualization '05* (2005), pp. 367–374.

[ZHT02] ZHOU J., HINZ M., TONNIES K.: Focal region-guided feature-based volume rendering. *Proc. of the First International Symposium 3D Data Processing Visualization and Transmission* (2002), 87–90.

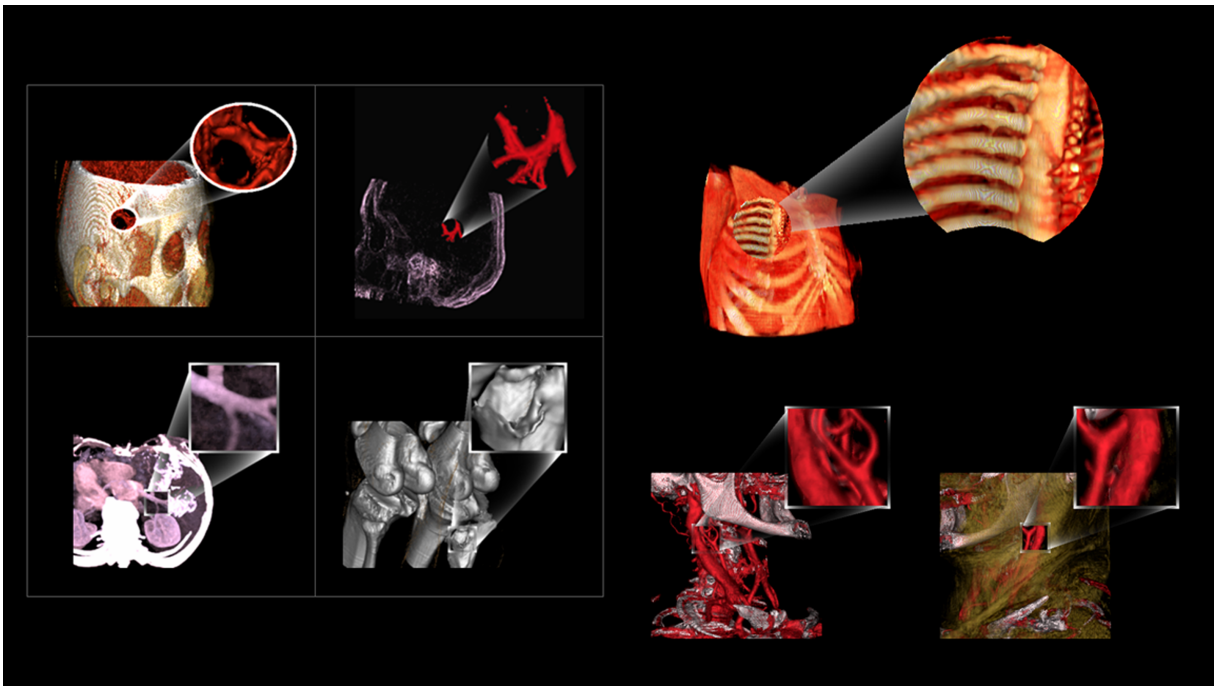

Figure 8: *(Color Plate) Contextual close-up with circular and rectangular ROIs: cerebral blood flow CT angiogram with contrast agent; MRI of liver with maximum intensity projection; CT scan of an anterior tibial osteotomy knee dataset; CT chest dataset with inside of rib cage; CT angiogram of the neck showing contrast-enhanced blood vessels of the right carotid artery and jugular vein.*